SAPHIRE "Recover Cut Sets" Editor: A Technique for Automating the Manipulation and Edition of PRA Cut Sets[†]

Curtis L. Smith and Richard D. Fowler
Idaho National Engineering Laboratory
Lockheed Martin Idaho Technologies
Idaho Falls, ID 83415-3850

ABSTRACT

The accident sequence cut sets generated for a probabilistic risk analysis generally require some manipulation to account for special modeling concerns. This paper presents a method of efficiently manipulating cut sets, specifically for the modeling concerns of (a) post-accident operator recovery actions, (b) common cause failure modeling, and (c) removal of mutually exclusive events. The method presented consists of logic rules that define a cut set search criteria and changes to be applied to the cut sets meeting the search criteria. While this method of cut set manipulation is demonstrated using the "Recover Cut Sets" editor in the SAPHIRE risk assessment computer code, it is proposed that this methodology could become a standard method for cut set manipulation.

KEYWORDS

probabilistic risk assessment, recovery actions, common cause failures, mutually exclusive events

---

[†] Work supported by the U.S. Nuclear Regulatory Commission, Office of Nuclear Regulatory Research, Under DOE Idaho Field Office Contract DE-AC07-94ID13223.

1. Introduction

The purpose of this paper is to present an overview of the SAPHIRE[1] probabilistic risk analysis (PRA) computer code ”Recover Cut Sets” editor. The SAPHIRE ”Recover Cut Sets” editor provides a means to develop logic rules that allow for the inclusion of recovery events or the modification or deletion of system or sequence cut sets. This rule-based editor has evolved, from a system used for adding recovery events to system or sequence cut sets, into a more powerful rule-based system that includes features for editing and manipulating cut sets. Consequently, the SAPHIRE “Recover Cut Set” editor can be used to automate advanced PRA techniques such as (1) the incorporation of component or system non-recovery events, (2) the inclusion of common-cause failure cut sets, and (3) the elimination of mutually exclusive events (e.g., restricted or impermissible combinations of events). Since many PRA reference texts (e.g., the *PRA Procedures Guide*[2]) address these three techniques, only a brief discussion of each will be provided. Although the discussion and examples deal with nuclear power plants, the ideas presented in this paper could be used to the advantage of other PRA applications.

An additional motivation for this paper is to present the PRA community with a method and syntax for modifying cut sets that could become standardized. While many PRA computer codes provide some limited cut set manipulation or deletion capabilities, it would be beneficial if a comprehensive set of cut set modification keywords were used universally by all PRA codes. Since the SAPHIRE code is the first widely available PRA code that contains a comprehensive list of cut set modification keywords, the authors venture to proffer these keywords to the PRA community to foster a standard method and syntax for cut set manipulation.

The remainder of this paper is centered on details when using the SAPHIRE “Recover Cut Sets” editor and related nomenclature. Specifically, Section 2 provides a discussion of the syntax used for creating rules. Section 3 shows the methodology used for incorporating non-recovery probabilities in PRA accident sequences. Section 4 shows an example of how to develop rules for handling common-cause failure modeling. Section 5 shows how rules can be created for removing mutually exclusive events. And finally, Section 6 presents concluding thoughts on the SAPHIRE “Recover Cut Sets” editor.

2. SAPHIRE “Recover Cut Sets” Editor

Structure for the “Recover Cut Sets” editor follows a format similar to that found in traditional programming languages (e.g., BASIC, FORTRAN, or C). As such, the ability exists to define "macros" and "if . . . then" type of structures. After creating or editing rules, SAPHIRE compiles the rules to check their validity. Alternatively, the rules can be entered in any word processor or text editor (that can output ASCII files) and then loaded directly into the SAPHIRE data base. Any errors or omissions in the rules are pointed out by the code. Also, while creating or editing a rule, initiating events, basic events, and recovery actions can all be directly added into the data base.

Rules created using the SAPHIRE “Recover Cut Sets” editor can be applied to cut sets for a particular PRA accident sequence, a single event tree, or all sequences in the PRA. Also, the rules can be applied to cut sets for a particular system (i.e., fault tree) or all systems in the PRA. The rules are entered in a "free-form" text editor within the SAPHIRE code. The structure of the

rules and the keywords that are available is discussed below. Table 1 contains a list of keywords and their definitions used in the SAPHIRE "Recover Cut Set" editor.

When applying a rule, SAPHIRE searches the list of either (a) system failure cut sets or (b) event tree accident sequence cut sets. These lists are searched for cut sets that match the search criteria defined in the rule. This searching process is a multi-step procedure.

1. If the first cut set in the list matches the search criteria in the first rule, the applicable modifications defined in the first rule are made to the cut set. Once a cut set qualifies for a search criterion it is not affected by any of the remaining rules.

2. If the first cut set in the list *does not* match the search criteria in the first rule, the second rule (SAPHIRE processes the rules from top to bottom) is evaluated.

3. After all the rules are evaluated for the first cut set, the second cut set in the list is evaluated, starting again with the first rule. This process is repeated until all cut sets are evaluated.

Special symbols used in the SAPHIRE "Recover Cut Set" editor are shown in Table 2. Using the boolean operators shown in Table 2, very complex expressions can be defined and used as the cut set search criteria in the rules. To demonstrate these operators, some examples of how the symbols can be used as cut set search criteria are:

1.	X	Basic event X appears in the cut set
2.	~X	Basic event X never occurs in the cut set
3.	/X	Success of event X appears in the cut set
4.	X * Y	Both events X and Y appear in the cut set
5.	X + Y	Either event X or Y appear in the cut set
6.	X*(Y + Z)	Either X and Y or X and Z appear in the cut set
7.	~X*Y	X does not appear, but Y does appear.

As previously mentioned, the general structure of the SAPHIRE “Recover Cut Set” editor follows the structure one would find in any traditional computer programming language. Shown below are two examples of the actual rule structure and syntax. Example A shows how the "if...then" rule structure can be used to check if a failure of an electric ac power bus event appears in a cut set. If the event does occur, the rule will multiply the cut set containing the event by an operator non-recovery probability named AC-BUS-REC via the "recovery" keyword. Example B is more complicated in that it shows the use of the "elsif" keyword. For this second example, the rule first checks to see if two diesel generator (DG) maintenance events appear in the same cut set. If they do, the cut set is deleted since plant Technical Specifications do not permit two diesel generators to be out of service simultaneously for routine maintenance. Or, if the cut set does not qualify under the first search criteria, the rule checks to see if the initiating events IE-LOSP and IE-SBO appear in the same cut set. If they do, the rule will remove the IE-LOSP initiating event, since a SBO cannot occur without a LOSP.

EXAMPLE A

```
| The "if...then" rule structure.
| This rule adds a recovery action AC-BUS-REC  when either
| electric bus B or C is failed.
if  AC-BUS-B + AC-BUS-C then
  recovery = AC-BUS-REC;
endif
```

EXAMPLE B

```
| The "if...then...elsif" structure.
| This rule deletes the cut set if both diesel generators are out for maintenance.
| Alternatively, if initiating events IE-LOSP and IE-SBO appear in the same
| cut set then delete the initiator IE-LOSP.
if (DG-1-MAINT * DG-2-MAINT) then
  DeleteRoot;
elsif  (init(IE-LOSP) * init(IE-SBO) then
  DeleteEvent = IE-LOSP;
endif
```

3. Recovery Actions.

In a PRA, cut sets for accident sequences are generated using the fault tree and event tree logic. Since most PRAs are analyzed in failure space, each cut set represents the minimal set of components or systems that have to fail (given a particular initiating event) in order to result in an undesired condition (e.g., core damage). As such, operator actions that could prevent the accident sequence from progressing to the point of an accident may not be specifically included

in the logic models.[††] To model the PRA accident sequences as accurately as practical, the analyst will typically want to apply recovery events (as appropriate) to the accident sequence cut sets.

The recovery events represent the probability that the operator or operators *fail* to successfully prevent the accident by restoring one or more of the failed components in the sequence cut sets. Consequently, the recovery events are frequently called the non-recovery probability events. Human reliability analysis is generally needed to quantify what these non-recovery probabilities will be for the particular accident sequences that are being modeled. Several PRA texts discuss the analysis technique of post-accident human reliability and recovery modeling.[3,4.5] The NUREG-1150 methodology[6] focused on many recovery events for system diagnosis and recovery. Examples of the non-recovery probability for various pressurized water reactor (PWR) systems[7] include: (a) non-recovery probability of ac power in 24 hours is 6.1×10^{-2}, (b) non-recovery probability of the auxiliary feedwater system by manual actuation is 2.7×10^{-3}, and (c) non-recovery of steam generator integrity by isolating the blowdown line is 3.4×10^{-3}.

To demonstrate the use of the SAPHIRE "Recover Cut Sets" editor, the example below shows how rules can be created to apply recovery actions (or non-recovery probability events) to specific cut sets in a particular sequence. As shown, the rule will add the NRAC-12HR operator recovery event (recovery of DGs within 12 hours) to those cut sets that contain LOSP as the

[††] In nuclear power plant PRAs, an accident is typically defined as damage to the reactor core.

sequence initiating event and failure of either DG A or DG B. Once this rule is saved in the PRA data base, SAPHIRE will evaluate all the sequence cut sets and then automatically include the recovery actions as specified in the recovery rule. The search criteria specified in the "if . . . then" line can be as complex as needed.

RECOVERY ACTIONS EXAMPLE

```
| Search on LOSP initiator and failure of diesel generators A or B.
if  init(LOSP) * (DG-A + DG-B)  then
   recovery = NRAC-12HR;
endif
```

4. Common-Cause Modeling.

Common cause failure modeling, one part of dependent failure analysis, is a standard PRA modeling technique. It attempts to model simultaneous failures of multiple components due to a single cause (i.e., a common mode failure). Common cause failures can frequently appear as an important failure event for redundant trains in PRA. Since the time of the WASH-1400[8] report development, many methods of common cause failure modeling have been proposed. Some of the more popular models include: (a) the Beta Factor method, (b) the Multiple Greek Letter method, (c) the Alpha Factor method, and (d) the Binomial Failure Rate method. A review of these models may be found in various PRA references.[9,10]

In most PRA codes, any of the previously mentioned common cause failure estimation models may be used. Three methods in which common cause modeling can be incorporated into a PRA using SAPHIRE include:

1. Manually editing previously generated cut sets to add new common cause failure cut sets. Given large numbers of cut sets, this process can be a very time-consuming and prone to error in that some cut sets may unintentionally be overlooked.

2. Rules can be created to search for cut sets containing specific groups of independent, random failure events. Because the target (i.e., the independent failures) is legitimate and is retained in the data base, the cut set is copied. It is the copied cut set that is then modified by replacing the independent events with a single common cause term.

3. Modifying the logic models to directly model common cause failures. Including common cause failures directly in the logic models is the method that is traditionally used during the fault tree development process.

Of the three methods above, the preferred method of common cause modeling will depend on your particular PRA and analysis needs. Some benefits of adding common cause failure events through the use of the SAPHIRE "Recover Cut Sets" editor include (a) rules that are automatically applied to the affected cut sets, (b) not having to make changes to logic models, and (c) the elimination of time-consuming manual editing of cut sets. But, one very important drawback to the use of "Recover Cut Sets" editor for common cause modeling is that groups of independent failures must be present in the cut sets in order for a rule to replace the independent failures with a single common cause failure event. Consequently, if probability or size truncation is used when generating cut sets (which is usually the case for full-scope PRAs), cut sets may be omitted that would be above the truncation limit after being modified by the rule.

A case on how the SAPHIRE “Recover Cut Sets” editor can be used to add common-cause events to the cut sets is shown in the following example. The example defines a search criteria that identifies the failure combination of two auxiliary feedwater pumps (pump A and pump B). If these two basic events are found in a cut set, a new cut set will be created that replaces the independent failures of the two pumps with a single common-cause basic event.

COMMON CAUSE MODELING EXAMPLE

```
| Pick cut sets that have combinations of AFW-PUMP-A and AFW-PUMP-B.
if  AFW-PUMP-A * AFW-PUMP-B  then
  | First make a copy of the original cut set
  CopyRoot;
  | Now remove the two independent failure events
  DeleteEvent = AFW-PUMP-A;
  DeleteEvent = AFW-PUMP-B;
  | Now add the CCF event
  AddEvent = AFW-PUMP-CCF;
endif
```

5. Mutually Exclusive Events

The term "mutually exclusive events" refers to two or more basic events that appear in a single cut set (either for systems or sequences) which logically should not appear together. Generally, mutually exclusive events should not appear together in the list of cut sets for one of two reasons.

1. Plant Technical Specifications or other operating restrictions may prevent two components from being out of service at the same time. An example is not allowing two AFW pumps to be simultaneously out of service for testing and maintenance.

2. Other general logic modeling concerns may lead the analyst to remove specific combinations of events. An example of this involves the practice of including multiple initiating events in the fault tree logic. Given this case, sequence cut sets can be generated that include multiple initiating events.

During the PRA logic modeling phase, the analyst may recognize that certain combinations of mutually exclusive events will appear just by knowing how the fault tree and event tree logic modeling was performed. However, some unrecognized mutually exclusive events may not be evident until the analyst generates and evaluates the system or sequence cut sets.

As an example of how fault tree logic modeling can produce mutually exclusive events, the fault tree shown in Figure 1 will be used. Generating cut sets for this fault tree will produce a cut set containing the two maintenance events DG-A-MAINT * DG-B-MAINT. If the plant Technical Specifications restrict both diesel generators from being in maintenance simultaneously while at power, this cut set is an example of mutually exclusive events. These mutually exclusive events would be handled by removing any cut sets that contain the mutually exclusive events (even if the cut set includes additional events). Several methods may exist that could perform this removal operation.

1. The cut sets can be manually edited to "weed-out" those with mutually exclusive events. This method is not efficient given large numbers of cut sets, and it may result in unintentionally overlooking affected cut sets.

2. Adding a complemented top event to event tree sequences that, when generating sequence cut sets, would remove impermissible events.

3. Modifying fault tree logic models (via logical NOT gates or complemented events) to effectively remove impermissible combinations of events.

4. Using the SAPHIRE “Recover Cut Set” editor to create rules to automatically remove those cut sets that contain impermissible combinations of events.

Using the SAPHIRE “Recover Cut Set” editor, the following example shows how a rule can be applied to remove a particular cut set from the cut set list. The example defines a “macro” called DGS-IN-MAINT as the search criteria. If the two diesel generators are in maintenance simultaneously then the cut set containing those two basic events is deleted.

MUTUALLY EXCLUSIVE EVENT EXAMPLE

```
| Define a macro that targets those cut sets that have combinations
| of two diesel generators out for maintenance.
DGS-IN-MAINT = DG-A-MAINT * DG-B-MAINT;
| Search for the maintenance events and then delete cut set.
if  DGS-IN-MAINT  then
| Delete the cut set
  DeleteRoot;
endif
```

6. Conclusions

This paper presents a method of manipulating PRA accident sequence cut sets using the SAPHIRE "Recover Cut Sets" editor. Examples are presented for the PRA modeling techniques of (a) post-accident operator recovery actions, (b) common cause failure modeling, and (c) the removal of mutually exclusive events. Usually, the rules provide an effective method for the PRA modeling techniques previously discussed, especially when considering the other potential options (e.g., manual cut set manipulation, logic model manipulations) presented.

The SAPHIRE "Recover Cut Sets" editor provides a useful method of cut set manipulation for both the inclusion of operator recovery actions and the removal of mutually exclusive events. Use of the "Recover Cut Sets" editor for common cause modeling may not be as useful because groups of independent failures must be present in the cut sets in order for a rule to replace the independent failures with a common cause failure event. Thus, it may be possible to unintentionally omit cut sets that would subsequently be above the truncation limit after being modified by the rule.

It is proposed that the PRA community could standardize on the keywords and keyword behavior presented in the paper for modifying cut sets system or sequence cut sets. The keywords used in SAPHIRE were developed with the intent that they would encompass all actions that an analyst would possibly need to perform during cut set manipulation. Consequently, it would be beneficial if these keywords were used universally by all PRA codes in order to further the idea of a PRA "convergence."

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for any third party's use, or the results of such use, of any information, apparatus, product or process disclosed in this report, or represents that its use by such third party would not infringe privately owned rights. The views expressed in this report are not necessarily those of the U.S. Nuclear Regulatory Commission.

Table 1. "Recover Cut Set" editor KEYWORDS used in SAPHIRE.

Keyword	Definition
if then	Indicates a search criteria is being specified.
elsif	Used to specify an alternative search criteria. Any number of "elsif" commands can be used within a rule.
else	Used to specify an action to be taken if all the search criteria are not met. The "else" keyword should be used as the last condition in the rule.
endif	Indicates the end of a particular rule.
always	Indicates that every cut set in the list that is being evaluated satisfies the search criteria.
init( )	Used in the search criteria to indicate that a sequence cut set has a particular initiating event.
recovery =	The recovery event specified is to be added to the cut set being evaluated (IRRAS keeps a record of all recovery events).
AddEvent =	The event specified is to be appended to the cut set being evaluated.
DeleteEvent=	The event specified is to be deleted from the cut set being evaluated.
NewCutset;	A new, empty cut set will be added to the list of cut sets. This new cut set then becomes the cut set that is being evaluated.
DeleteRoot;	The original cut set (i.e., that cut set that satisfied the search criteria) is to be deleted.
CopyCutset;	The cut set being evaluated is to be copied and added to the list of cut sets. This copied cut set then becomes the cut set that is being evaluated.
CopyRoot;	The original cut set (i.e., the cut set that satisfied the search criteria) is to be copied. This copied cut set then becomes the cut set that is being evaluated.

Table 2. Additional “Recover Cut Sets” editor symbols used in SAPHIRE.

Symbol	Usage
|	The "pipe" symbol is used to begin comment lines.
~	Logical operator meaning "never" or "not present." This symbol is used when it is desired to test on an event not appearing in a cut set.
*	Logical AND (i.e., intersection) operator.
+	Logical OR (i.e., union) operator.
/	Signifies the complement of an event (e.g., /A = 1 - A).
( )	Parentheses may be used to group terms in a logical expression.

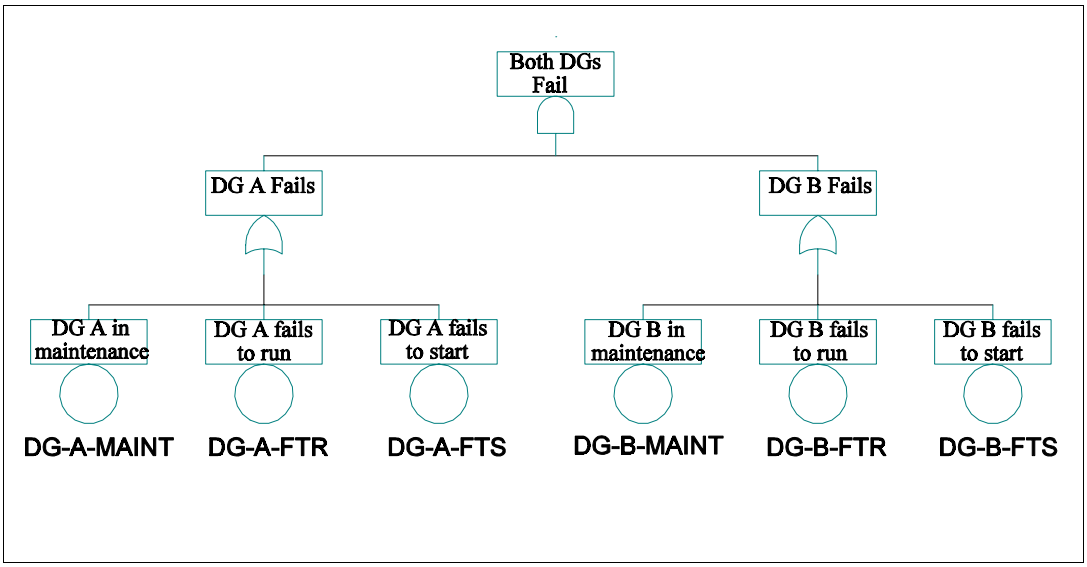

Figure 1. Example fault tree that contains mutually exclusive events.

7. References

1. K. D. Russell et al., *Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 5.0*, NUREG/CR-6116, Vols. 1 through 10, 1994.

2. J. W. Hickman, PRA Procedures Guide: A Guide to the Performance of Probabilistic Risk Assessments for Nuclear Power Plants, American Nuclear Society and Institute of Electrical and Electronic Engineers, NUREG/CR-2300, Vols. 1 and 2, January 1983.

3. Swain, A. D., and H. E. Guttman, *Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications*, NUREG/CR-1278, SAND80-0200, Sandia National Laboratories, August 1983.

4. Bell, B. J. and A. D. Swain, *A Procedure for Conducting a Human Reliability Analysis for Nuclear Power Plants - Final Report*, NUREG/CR-2254, SAND81-1655, May 1983.

5. Embry, D. E. et al., *SLIM-MAUD: An Approach to Assessing Human Error Probabilities Using Structured Expert Judgment*, U.S. Nuclear Regulatory Commission, NUREG/CR-3518, Washington D.C.

6. Ericson, D. M. et al., *Analysis of Core Damage Frequency: Internal Events Methodology*, NUREG/CR-4550, Vol. 1, Rev. 1, Sandia National Laboratories, January 1990.

7. Bertucio, R. C. and J. A. Julius, *Analysis of Core Damage Frequency: Surry, Unit 1 Internal Events*, NUREG/CR-4550, Vol. 3, Rev.1, Parts 1 and 2, April 1990.

8. Reactor Safety Study, *An Assessment of Accident Risk in U.S. Commercial Nuclear Power Plants*, U.S. Nuclear Regulatory Commission, WASH-1400, 1975.

9. Modarres, M., *What Every Engineer Should Know About Reliability and Risk Analysis*, Marcel Dekker, Inc., New York, NY, 1993.

10. U.S. Nuclear Regulatory Commission, *A Review of NRC Staff Uses of Probabilistic Risk Assessment*, NUREG-1489, March 1994.